AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1. (Currently amended) A method to efficiently realize class initialization barriers in a multitasking virtual machine, wherein class loading always takes place before class initialization, and wherein a class initialization barrier guarantees that a class is initialized before the class is first used by a program, comprising:

associating a shared runtime representation of the class with a task class mirror table that comprises at least one entry per-task, including an initialized entry, for a plurality of tasks, wherein each entry holds either a null pointer value or a non-null pointer to a task class mirror object, wherein all entries of a task mirror table that hold a non-null pointer value and that are associated with a same task hold a pointer to a same task class mirror object, wherein the task class mirror object holds a task private representation of the class for that task, wherein each task is associated with a unique integer value, wherein the unique integer value is used to compute a byte-offset from a beginning of task class mirror tables that can be used to retrieve from the initialized entry of any task class mirror table the pointer to the task class mirror object, wherein a computed byte-offset to the initialized entry is stored in a descriptor of a plurality of threads executing on behalf of a corresponding task;

using the initialized entry of a task in the task class mirror table to determine whether this task has initialized the class associated with the task class mirror table~~; and~~, which involves:

examining the initialized entry of the task in the task class mirror table associated with the class in order to determine if that task has initialized the class, wherein the byte-offset to the initialized entry from the beginning of the task class mirror table is obtained from the descriptor of a thread performing an examination on behalf of the task; and

initializing the class by the task if the class is not already initialized, wherein a null pointer stored at the initialized entry indicates that the class has not initialized the task, wherein a non-null pointer value indicates that the class has been initialized; and

accessing the task class mirror object associated to a particular task.

2. (Cancelled)

3. (Currently amended) The method of ~~clam 2~~claim 1, further comprising:

creating the task class mirror table and associating the task class mirror table with the shared runtime representation of the class upon creation of the shared runtime representation of the class; and

setting all entries of the task class mirror table to the null pointer value.

4. (Cancelled)

5. (Currently amended) The method of ~~claim 4~~Claim 1, further comprising:

upon completion of initialization of the class by the task, setting the initialized entry of the task class mirror table associated with the class to the task class mirror object that holds a representation of the class that is private to the task; and setting this task class mirror object to a fully initialized state.

6. (Original) The method of claim 5, wherein task class mirror tables associated with classes that have a non-empty initialization function includes one resolved entry per-task in addition to one initialized entry per-task, for the plurality of tasks.

7. (Original) The method of claim 6, wherein task class mirror tables associated with classes that have an empty initialization function include one resolved entry per-task in addition to an initialized entry per-task, for the plurality of tasks.

8. (Original) The method of claim 7, further comprising:

upon loading any class by the task, creating the task class mirror object that holds the task private representation of the class;

setting the task class mirror object's state to loaded; and

assigning the task class mirror object's pointer to a resolved entry of the task class mirror table associated with the class for that task.

9. (Original) The method of claim 8,

wherein the task class mirror table is arranged so that the resolved entry and the initialized entry for the task are consecutive; and

wherein the byte-offset to the resolved entry can be computed from the byte-offset to the initialized entry for a same task by adding a size, expressed in number of bytes, of the pointer to the task class mirror object.

10. (Original) The method of claim 8,

wherein the task class mirror table is arranged so that the resolved entry and the initialized entry for the task are separated by half of a total number of entries in the task class mirror table; and

wherein the byte-offset to the resolved entry can be computed from the byte-offset to the initialized entry for a same task by adding a size, expressed in number of bytes, of half the total number of entries in the task class mirror table.

11. (Original) The method of claim 8, wherein the resolved entry of the task class mirror table associated with the class is used in cases where testing for class initialization is unneeded but access to a task-private part of the class is required when the class has been loaded but not fully initialized.

12. (Original) The method of claim 6,

wherein task class mirror tables associated with classes that have an empty initialization function have a single entry per task; and

wherein the single entry per task is the initialized entry for that task.

13. (Original) The method of claim 12, further comprising:

upon loading the class that has the non-empty initialization function by the task, creating the task class mirror object that holds the task private representation of the class;

setting the task class mirror object's state to loaded; and

assigning the task class mirror object's pointer to a resolved entry of the task class mirror table associated with the class for that task.

14. (Original) The method of claim 13,

wherein the task class mirror table is arranged so that the resolved entry and the initialized entry for the task are separated by half of a total number of entries in the task class mirror table; and

wherein the byte-offset to the resolved entry can be computed from the byte-offset to the initialized entry for a same task by adding a size, expressed in number of bytes, of half the total number of entries in the task class mirror table.

15. (Original) The method of claim 14, wherein the resolved entry of task class mirror tables associated with classes that have the non-empty initialization function is used when accessing a task-private part of the class without testing for class initialization is necessary and the task has loaded but not fully initialized the class.

16. (Original) The method of claim 12, further comprising:

upon loading of the class that has the empty initialization function by the task, creating the task class mirror object that holds the task private representation of the class;

setting the task class mirror object's state to fully initialized; and

assigning the task class mirror object's pointer to the initialized entry of the task class mirror table associated with the class for that task.

17. (Currently amended) A computer-readable storage medium storing instructions that when executed by a computer cause the computer to perform a method to efficiently realize class initialization barriers in a multitasking virtual machine, wherein class loading always takes place before class initialization, and wherein a class initialization barrier guarantees that a class is initialized before the class is first used by a program, comprising:

associating a shared runtime representation of the class with a task class mirror table that comprises at least one entry per-task, including an initialized entry, for a plurality of tasks, wherein each entry holds either a null pointer value or a non-null pointer to a task class mirror object, wherein all entries of a task mirror table that hold a non-null pointer value and that are associated with a same task hold a pointer to a same task class mirror object, wherein the task class mirror object holds a task private representation of the class for that task, wherein each task is associated with a unique integer value, wherein the unique integer value is used to compute a byte-offset from a beginning of task class mirror tables that can be used to retrieve from the initialized entry of any task class mirror table the pointer to the task class mirror object, wherein a computed byte-offset to the initialized entry is stored in a descriptor of a plurality of threads executing on behalf of a corresponding task;

using the initialized entry of a task in the task class mirror table to determine whether this task has initialized the class associated with the task class mirror table~~; and~~, which involves:

examining the initialized entry of the task in the task class mirror table associated with the class in order to determine if that task has initialized the class, wherein the byte-offset to the initialized entry from the beginning of the task class mirror table is obtained from the descriptor of a thread performing an examination on behalf of the task ; and

initializing the class by the task if the class is not already initialized, wherein a null pointer stored at the initialized entry indicates that the class has not initialized the task, wherein a non-null pointer value indicates that the class has been initialized; and

accessing the task class mirror object associated to a particular task.

18. (Cancelled)

19. (Currently amended) The computer-readable storage medium of ~~claim 18~~claim 17, the method further comprising:

creating the task class mirror table and associating the task class mirror table with the shared runtime representation of the class upon creation of the shared runtime representation of the class; and

setting all entries of the task class mirror table to the null pointer value.

20. (Cancelled)

21. (Currently amended) The computer-readable storage medium of ~~claim 20~~claim 17, the method further comprising:

upon completion of initialization of the class by the task, setting the initialized entry of the task class mirror table associated with the class to the task class mirror object that holds a representation of the class that is private to the task; and

setting this task class mirror object to a fully initialized state.

22. (Original) The computer-readable storage medium of claim 21, wherein task class mirror tables associated with classes that have a non-empty initialization function includes one resolved entry per-task in addition to one initialized entry per-task, for the plurality of tasks.

23. (Original) The computer-readable storage medium of claim 22, wherein task class mirror tables associated with classes that have an empty initialization function includes one resolved entry per-task in addition to an initialized entry per-task, for the plurality of tasks.

24. (Original) The computer-readable storage medium of claim 23, the method further comprising:

upon loading any class by the task, creating the task class mirror object that holds the task private representation of the class;

setting the task class mirror object's state to loaded; and

assigning the task class mirror object's pointer to a resolved entry of the task class mirror table associated with the class for that task.

25. (Original) The computer-readable storage medium of claim 24,

wherein the task class mirror table is arranged so that the resolved entry and the initialized entry for the task are consecutive; and

wherein the byte-offset to the resolved entry can be computed from the byte-offset to the initialized entry for a same task by adding a size, expressed in number of bytes, of the pointer to the task class mirror object.

26. (Original) The computer-readable storage medium of claim 24,

wherein the task class mirror table is arranged so that the resolved entry and the initialized entry for the task are separated by half of a total number of entries in the task class mirror table; and

wherein the byte-offset to the resolved entry can be computed from the byte-offset to the initialized entry for a same task by adding a size, expressed in number of bytes, of half the total number of entries in the task class mirror table.

27. (Original) The computer-readable storage medium of claim 24, wherein the resolved entry of the task class mirror table associated with the class is used in cases where testing for class initialization is unneeded but access to a task-private part of the class is required when the class has been loaded but not fully initialized.

28. (Original) The computer-readable storage medium of claim 22,

wherein task class mirror tables associated with classes that have an empty initialization function have a single entry per task; and

wherein the single entry per task is the initialized entry for that task.

29. (Original) The computer-readable storage medium of claim 28, the method further comprising:

upon loading the class that has the non-empty initialization function by the task, creating the task class mirror object that holds the task private representation of the class;

setting the task class mirror object's state to loaded; and

assigning the task class mirror object's pointer to a resolved entry of the task class mirror table associated with the class for that task.

30. (Original) The computer-readable storage medium of claim 29,

wherein the task class mirror table is arranged so that the resolved entry and the initialized entry for the task are separated by half of a total number of entries in the task class mirror table; and

wherein the byte-offset to the resolved entry can be computed from the byte-offset to the initialized entry for a same task by adding a size, expressed in number of bytes, of half the total number of entries in the task class mirror table.

31. (Original) The computer-readable storage medium of claim 30, wherein the resolved entry of task class mirror tables associated with classes that have the non-empty initialization function is used when accessing a task-private part of the class without testing for class initialization is necessary and the task has loaded but not fully initialized the class.

32. (Original) The computer-readable storage medium of claim 28, the method further comprising:

upon loading of the class that has the empty initialization function by the task, creating the task class mirror object that holds the task private representation of the class;

setting the task class mirror object's state to fully initialized; and

assigning the task class mirror object's pointer to the initialized entry of the task class mirror table associated with the class for that task.

33. (Currently amended) An apparatus to efficiently realize class initialization barriers in a multitasking virtual machine, wherein class loading always takes place before class initialization, and wherein a class initialization barrier guarantees that a class is initialized before the class is first used by a program, comprising:

an associating mechanism that is configured to associated a shared runtime representation of the class with a task class mirror table that comprises at least one entry per-task, including an initialized entry, for a plurality of tasks, wherein each entry holds either a null pointer value or a non-null pointer to a task class mirror object, wherein all entries of a task mirror table that hold a non-null pointer value and that are associated with a same task hold a pointer to a same task class mirror object, wherein the task class mirror object holds a task private representation of the class for that task, wherein each task is associated with a unique integer value, wherein the unique integer value is used to compute a byte-offset from a beginning of task class mirror tables that can be used to retrieve from the initialized entry of any task class mirror table the pointer to the task class mirror object, wherein a computed byte-offset to the initialized entry is stored in a descriptor of a plurality of threads executing on behalf of a corresponding task;

a determining mechanism that is configured to use the initialized entry of a task in the task class mirror table to determine whether this task has initialized the class associated with the task class mirror table~~; and~~, which involves:

examining the initialized entry of the task in the task class mirror table associated with the class in order to determine if that task has initialized the class, wherein the byte-offset to the initialized entry from the beginning of the task class mirror table is obtained from the descriptor of a thread performing an examination on behalf of the task; and

initializing the class by the task if the class is not already initialized, wherein a null pointer stored at the initialized entry indicates that the class has not initialized the task, wherein a non-null pointer value indicates that the class has been initialized; and

an accessing mechanism that is configured to access the task class mirror object associated to a particular task.

34. (Cancelled)

35. (Currently amended) The apparatus of ~~claim 34~~claim 33, further comprising:

a creating mechanism that is configured to create the task class mirror table and associating the task class mirror table with the shared runtime representation of the class upon creation of the shared runtime representation of the class; and

a setting mechanism that is configured to set all entries of the task class mirror table to the null pointer value.

36. (Cancelled)

37. (Currently amended) The apparatus of ~~claim 36~~claim 33, wherein the setting mechanism is further configured to set the initialized entry of the task class mirror table associated with the class to the task class mirror object that holds a representation of the class that is private to the task; and

wherein the setting mechanism is further configured to set this task class mirror object to a fully initialized state.

38. (Original) The apparatus of claim 37, wherein task class mirror tables associated with classes that have a non-empty initialization function includes one resolved entry per-task in addition to one initialized entry per-task, for the plurality of tasks.

39. (Original) The apparatus of claim 38, wherein task class mirror tables associated with classes that have an empty initialization function includes one resolved entry per-task in addition to an initialized entry per-task, for the plurality of tasks.

40. (Original) The apparatus of claim 39,

wherein the creating mechanism is further configured to create the task class mirror object that holds the task private representation of the class;

wherein the setting mechanism is further configured to set the task class mirror object's state to loaded; and

further comprising an assigning mechanism that is configured to assign the task class mirror object's pointer to a resolved entry of the task class mirror table associated with the class for that task.

41. (Original) The apparatus of claim 40,

wherein the task class mirror table is arranged so that the resolved entry and the initialized entry for the task are consecutive; and

wherein the byte-offset to the resolved entry can be computed from the byte-offset to the initialized entry for a same task by adding a size, expressed in number of bytes, of the pointer to the task class mirror object.

42. (Original) The apparatus of claim 40,

wherein the task class mirror table is arranged so that the resolved entry and the initialized entry for the task are separated by half of a total number of entries in the task class mirror table; and

wherein the byte-offset to the resolved entry can be computed from the byte-offset to the initialized entry for a same task by adding a size, expressed in number of bytes, of half the total number of entries in the task class mirror table.

43. (Original) The apparatus of claim 40, wherein the resolved entry of the task class mirror table associated with the class is used in cases where testing for class initialization is unneeded but access to a task-private part of the class is required when the class has been loaded but not fully initialized.

44. (Original) The apparatus of claim 38,

wherein task class mirror tables associated with classes that have an empty initialization function have a single entry per task; and

wherein the single entry per task is the initialized entry for that task.

45. (Original) The apparatus of claim 44,

wherein the creating mechanism is further configured to create the task class mirror object that holds the task private representation of the class;

wherein the setting mechanism is further configured to set the task class mirror object's state to loaded; and

further comprising an assigning mechanism that is configured to assign the task class mirror object's pointer to a resolved entry of the task class mirror table associated with the class for that task.

46. (Original) The apparatus of claim 45,

wherein the task class mirror table is arranged so that the resolved entry and the initialized entry for the task are separated by half of a total number of entries in the task class mirror table; and

wherein the byte-offset to the resolved entry can be computed from the byte-offset to the initialized entry for a same task by adding a size, expressed in number of bytes, of half the total number of entries in the task class mirror table.

47. (Original) The apparatus of claim 46, wherein the resolved entry of task class mirror tables associated with classes that have the non-empty initialization function is used when accessing a task-private part of the class without testing for class initialization is necessary and the task has loaded but not fully initialized the class.

48. (Original) The apparatus of claim 44,

wherein the creating mechanism is further configured to create the task class mirror object that holds the task private representation of the class;

wherein the setting mechanism is further configured to set the task class mirror object's state to fully initialized; and

further comprising an assigning mechanism that is configured to assign the task class mirror object's pointer to the initialized entry of the task class mirror table associated with the class for that task.